Programación de Microservicios con Spring Boot y Red Hat SSO

Nivel Avanzado

Instructor: Carlos Carreño
ccarrenovi@gmail.com

Modulo 6 Integración de Spring Boot con Red Hat SSO

Objetivo: Introducción a Red Hat SSO y su integración con el desarrollo de aplicaciones con Spring Boot

Duración: 1h

Introducción a Spring Security con OAuth2

- **OAuth2** es un protocolo estándar de autorización usado para obtener acceso limitado a recursos protegidos sin compartir las credenciales del usuario.
- En Spring Boot/Security permite validar tokens JWT emitidos por servidores de autenticación como Red Hat SSO/Keycloak.

Componentes de OAuth2 con Spring Security

Componente	Rol
Resource Server (Spring Boot App)	Expone APIs protegidas por OAuth2.
Authorization Server (Keycloak/Red Hat SSO)	Emite tokens JWT.
Client App (Opcional)	Aplicación frontend que pide tokens.

Dependencias OAuth2 y Spring Security

```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-oauth2-resource-server</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

Configuración

```
spring.security.oauth2.resourceserver.jwt.issuer-uri=http://localhost:8080/realms/laboratorio-realm
```

Protegiendo el Controlador

```
@RestController
@RequestMapping("/api/secure")
public class SecureController {

    @GetMapping
    public String securedEndpoint() {
        return "Acceso concedido: token JWT válido!";
    }
}
```

Configuración desde un Bean

```java
import org.springframework.context.annotation.Bean;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.web.SecurityFilterChain;

@Configuration
public class SecurityConfig {

    @Bean
    SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        http
            .authorizeHttpRequests(auth -> auth
                .requestMatchers("/api/secure").authenticated()
                .anyRequest().permitAll()
            )
            .oauth2ResourceServer(oauth2 -> oauth2.jwt());

        return http.build();
    }
}
```

Configuración del application.properties para OIDC

```
spring.security.oauth2.client.registration.keycloak.client-id=springboot-app
spring.security.oauth2.client.registration.keycloak.client-secret=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
spring.security.oauth2.client.registration.keycloak.scope=openid,profile,email
spring.security.oauth2.client.registration.keycloak.authorization-grant-type=authorization_code
spring.security.oauth2.client.registration.keycloak.redirect-uri=http://localhost:8081/login/oauth2/code/keycloak

spring.security.oauth2.client.provider.keycloak.issuer-uri=http://localhost:8080/realms/laboratorio-realm
spring.security.oauth2.client.provider.keycloak.user-name-attribute=preferred_username
```

continuación

Propiedad	Significado
client-id	ID del cliente registrado en Keycloak.
client-secret	Secreto del cliente (para aplicaciones confidenciales).
scope	Alcance de OIDC (siempre incluir `openid`).
authorization-grant-type	Tipo de flujo OIDC (usualmente `authorization_code`).
redirect-uri	URL a la que Keycloak devolverá el token después del login.
issuer-uri	Dirección base de tu Realm en Keycloak.
user-name-attribute	Campo del token JWT que representa el usuario (`preferred_username`).

Uso de Spring Boot Starter OAuth2 Resource Server

- Esta opción es para APIs backend (sin login), sólo validando tokens JWT.

```
spring.security.oauth2.resourceserver.jwt.issuer-uri=http://localhost:8080/realms/laboratorio-realm
```

Verificación de tokens JWT automáticamente

- En **Spring Boot**, la verificación de tokens JWT con OAuth2 / OIDC es **automática** cuando configuras el proyecto como un **Resource Server** o **OIDC Client**.

✓Firma JWT con la clave pública de Keycloak (obtenida de .well-known/openid-configuration).

✓Vigencia (exp, nbf, iat).

✓Emisor (iss) coincide con issuer-uri.

✓Audience (aud) si se configura (opcional).

✓Decodificación de claims (sub, email, roles, etc.).

Lab

- Ejercicio práctico: Microservicio protegido por Red Hat SSO